Program 2 Report

Erik Mitchell

Student ID: 00001581305

Rank: 20, Score: 0.6503

Department of Computer Science & Engineering, Santa Clara University

CSEN 342: Deep Learning

Dr. David C. Anastasiu

February 20th, 2025

1. Introduction

The goal of this program project is to train a detection model using transfer learning and modify the hyperparameters to produce the best results on the given validation set. The given dataset consists of images depicting car traffic in intersections. We retrain a pre-trained detection model to only detect three kinds of objects: car, medium truck, and large truck. This report includes an analysis of the given dataset, describes the chosen pre-trained model, and explains the methodology and experiments that give great detection results on the validation set.

2. Methodology

2.1. Detection Model

For this project, I have chosen to finetune a YOLO (You Only Look Once) model. YOLO is a family of computer vision models that are very popular for solving modern detection problems. YOLO computes both object identification and classification within a single stage with strong accuracy and speed. The single stage consists of input images being segmented into a coarse grid where the model then directly predicts the class label and a few candidate boxes for each grid cell.

Since the publishing of the first YOLO model, the base framework has been iterated upon to make up a family of models. The latest model, as published by the Ultralytics team, is YOLO11. YOLO11 provides cutting-edge accuracy, speed, and efficiency through improvements in the architecture and training methods from previous YOLO versions. For this project, I try fine-tuning two YOLO11 pre-trained detection models: YOLO11n and YOLO11x. Both models are trained on the COCO8 dataset. Further details on the models can be found in Table 1.

	Size (pixels)	mAP 50-95	Speed on T4 TensorRT10 (ms)	Parameters (M)	FLOPs (B)
YOLO11n	640	39.5	1.5 ± 0.0	2.6	6.5
YOLO11x	640	54.7	11.3 ± 0.2	56.9	194.9

Table 1. YOLO11n and YOLO11x model details.

3. Experiments

3.1. Datasets

We are given three datasets for this project. Figure 1 shows the tree diagram for these datasets.

|-- [DIR] test
        |-- [DIR] images
                |-- [FILE] **image_id**.jpeg

```
|-- [DIR] train
        |-- [DIR] images
                |-- [FILE] image_id.jpeg
        |-- [FILE] labels.txt
|-- [DIR] val
        |-- [DIR] images
                |-- [FILE] image_id.jpeg
        |-- [FILE] labels.txt
```

Figure 1. Tree diagram of given datasets.

The directories named **images** contain all of the images for the dataset. Each image file, represented by the moniker **image_id**, is named based on the id of the image, starting with 00001.jpeg. The train directory contains 14,000 training images and the test and val directories each contain 2,000 images for testing and validation respectively.

3.2. Performance Metric

The metric used in the experiments to determine the performance of the retrained model is mean average precision (mAP).

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{|TP_c|}{|FP_c| + |TP_c|}$$

Not only is this metric used by our submission format to determine our leaderboard rankings, but it is also the same metric used to evaluate the YOLO models, as seen in Table 1. This means that we can directly analyze the performance of the base and fine tuned YOLO models on the given dataset using a consistent metric.

3.3. Dataset Formatting

As described in the datasets section (3.1), the training and validation datasets both have a labels.txt file that contains all of the bounding boxes information for all of the images within each dataset. This formatting is slightly different from the required dataset formatting that YOLO requires, which is a label.txt file for each image. Due to this difference, the code modifies the given dataset into a new dataset, called custom_dataset, that contains the modified files to fit the required format. There is also code after inference to reformat the predictions from the YOLO format back to the format needed for CLP submission. Comments within the code should describe the process clearly.

3.4. Tuning Hyperparameters

In order to improve the accuracy of the retrained YOLO model during our experiments, we modify the hyperparameters of the model. In these experiments, the number of parameters, number of epochs, batch size, and optimizer type were the hyperparameters that were modified.

4. Results

4.1. YOLO11n Model

On the YOLO 11n model, the following hyperparameters were used: 20 epochs, SGD optimizer, and batch size 16. On submission, this model received .6503 mAP (on 50%).

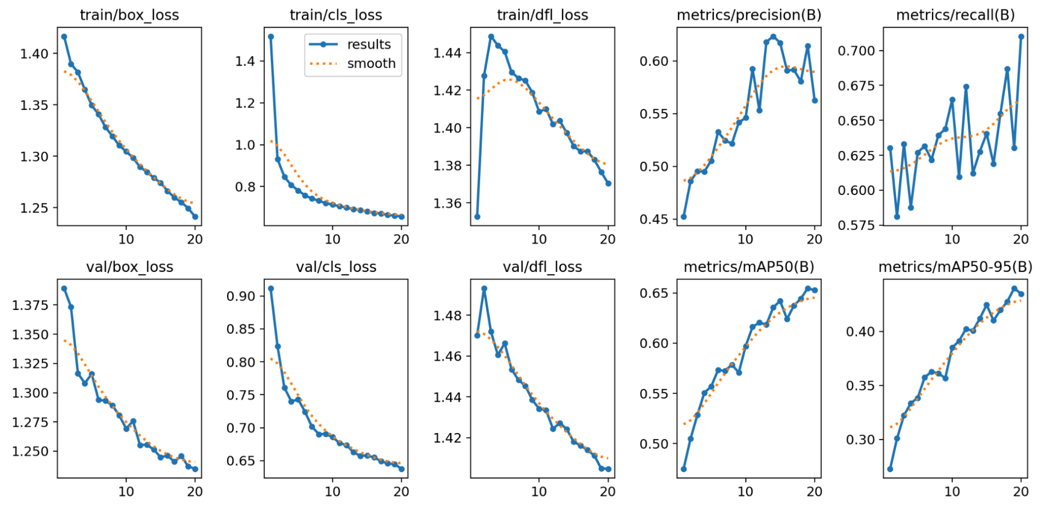

Figure 2. Training and validation loss and metrics for YOLO11n model

4.2. YOLO11x Model

On the YOLO11x model, the following hyperparameters were used: 100 epochs, Adam optimizer, and batch size 4. On submission, this model received .6073 mAP (on 50%).

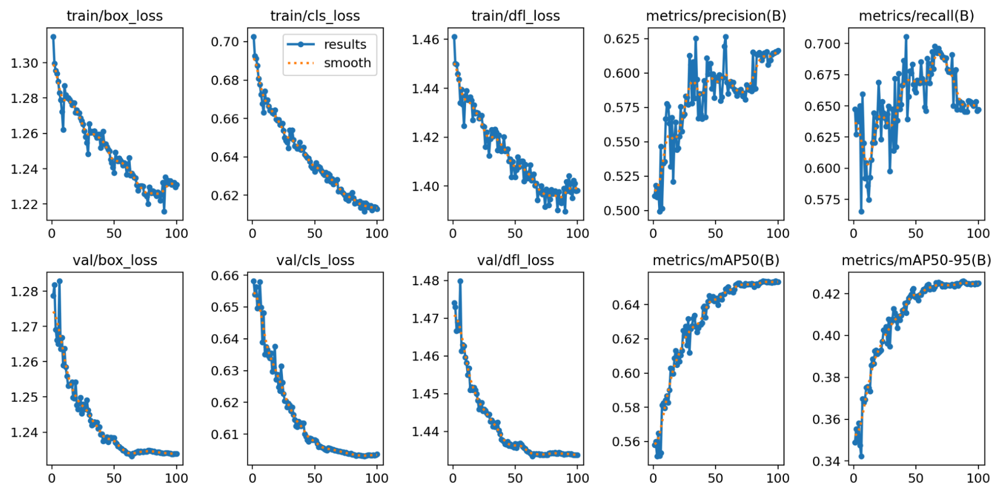

Figure 3. Training and validation loss and metrics for YOLO11x model

From the results, it's clear that having a larger model with more parameters, like the YOLO11x model, does not necessarily mean that it will perform better than a smaller model. It's also clear that the number of epochs has a large impact on how well the model will perform. The YOLO11n model seems like it could have performed even better with more training, as the bias seems low, and the YOLO11x model seems to have trained for too long, as the loss begins to increase at around 70 epochs. The loss increasing seems to indicate that the model started overfitting, indicating high variance. The smaller batch size used for the YOLO11x model may have introduced noise that resulted in worse performance. The SGD optimizer may have been the better function compared to Adam in this case, although more experiments would need to be done to confirm this.

5. Conclusion

YOLO is a strong computer vision model that can solve a variety of problems. By fine tuning the model, it was able to specifically detect the objects that I specified. From the multiple experiments, it is clear that the hyperparameters of the model significantly alter the performance. Finetuning is a strong skill that I hope to build upon for future deep learning projects.